# GCP-SR

December 11, 2017

```
In [1]: import tensorflow as tf
        import time,sys,os,math,random,itertools,glob,cv2
        from datetime import timedelta
        from sklearn.utils import shuffle
        import pandas as pd
        import numpy as np
        from datetime import timedelta
        from sklearn.model_selection import train_test_split,ShuffleSplit
        os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
        #Adding Seed so that random initialization is consistent
        from numpy.random import seed
        seed(2)
        from tensorflow import set_random_seed
        import matplotlib.pyplot as plt
        from tensorflow.python.framework import ops
        #get_ipython().run_line_magic('matplotlib', 'inline')
        set_random_seed(2)
        #final_project_path = r"M:\Course stuff\Fall 17\CMPS 242\final project"
        #os.chdir(final_project_path)



        def drawProgressBar(percent, barLen = 50):
                sys.stdout.write("\r")
                progress = ""
                for i in range(barLen):
                        if i<int(barLen * percent):
                                progress += "="
                        else:
                                progress += " "
                sys.stdout.write("[ %s ] %.2f%%" % (progress, percent * 100))
                sys.stdout.flush()



        imp_labels = ['yes', 'no', 'up', 'down', 'left', 'right', 'on', 'off', 'stop', 'go', 'si
        def load_train(train_path):
                images = []
        classes = []
        path = train_path
        file_names = os.listdir(os.path.join(os.getcwd(),train_path))
        counter = 1
        print("Creating Classes, reading images and breaking things ...\n")
        for file in file_names:
                drawProgressBar(counter/len(file_names))
                #print(file)
                classes.append(file.split("_")[0])
                image = cv2.imread(os.path.join(os.getcwd(),train_path,file))
                image = image.astype(np.float32)
                image = np.multiply(image, 1.0/255.0) #normalizing the pixel intensities
                images.append(image)
                counter += 1
        print("\nDone!")
        images = np.array(images)
        #classes now has all the labels. order preserved
        #but we need the classes to be floats/ints so lets map the shit out of them
        for i in range(len(classes)):
                if classes[i] not in imp_labels:
                        classes[i] = 'unkown'
        d = {ni:indi for indi, ni in enumerate(set(classes))}
        classes = [d[ni] for ni in classes]
        classes = np.array(classes)
        n_values = np.max(classes)+1
        classes = np.eye(n_values)[classes]
        #classes = np.eye(n_values)[classes.reshape(-1)]
        print("\nDone!")
        print("\n images shape: {}, labels shape: {}".format(images.shape,classes.shape)
        return (images,classes)#(train_x,train_y,test_x,test_y)


def split_data(images, labels,test_size = 0.2, random_state = 7, shuffle = False):
        return(train_test_split(images,labels,test_size = test_size,random_state = rando


def random_mini_batches(X, Y, mini_batch_size = 64, seed = 0):

        m = X.shape[0]                  # number of training examples
        mini_batches = []
        np.random.seed(seed)

        # Step 1: Shuffle (X, Y)
        permutation = list(np.random.permutation(m))
        shuffled_X = X[permutation,:,:,:]
        shuffled_Y = Y[permutation,:]

        # Step 2: Partition (shuffled_X, shuffled_Y). Minus the end case.
        num_complete_minibatches = math.floor(m/mini_batch_size) # number of mini batche
        for k in range(0, num_complete_minibatches):
                mini_batch_X = shuffled_X[k * mini_batch_size : k * mini_batch_size + mi
                mini_batch_Y = shuffled_Y[k * mini_batch_size : k * mini_batch_size + mi
                mini_batch = (mini_batch_X, mini_batch_Y)
                mini_batches.append(mini_batch)

        # Handling the end case (last mini-batch < mini_batch_size)
        if m % mini_batch_size != 0:
                mini_batch_X = shuffled_X[num_complete_minibatches * mini_batch_size : m
                mini_batch_Y = shuffled_Y[num_complete_minibatches * mini_batch_size : m
                mini_batch = (mini_batch_X, mini_batch_Y)
                mini_batches.append(mini_batch)

        return mini_batches


def create_placeholders(n_H0, n_W0, n_C0, n_y):

        X = tf.placeholder(shape = [None, n_H0, n_W0, n_C0],dtype = tf.float32)
        Y = tf.placeholder(shape = [None, n_y],dtype = tf.float32)

        return X, Y


def initialize_parameters():

        tf.set_random_seed(1)                              # so that your "random" numbe


        W1 = tf.get_variable("W1",[7, 7, 3, 8], initializer = tf.contrib.layers.xavier_i
        W2 = tf.get_variable("W2",[5, 5, 8, 16], initializer = tf.contrib.layers.xavier_
        W3 = tf.get_variable("W3",[3, 3, 16, 8], initializer = tf.contrib.layers.xavier_
        W4 = tf.get_variable("W4",[2, 2, 8, 4], initializer = tf.contrib.layers.xavier_i
        W5 = tf.get_variable("W5",[2, 2, 4, 4], initializer = tf.contrib.layers.xavier_i


        parameters = {"W1": W1,
                                "W2": W2,
                                "W3":W3,
                                "W4":W4,
                                "W5":W5}

        return parameters

def forward_propagation(X, parameters):

        W1 = parameters['W1']
        W2 = parameters['W2']
        W3 = parameters['W3']
        W4 = parameters['W4']
        W5 = parameters['W5']
        with tf.device('/device:GPU:0'):

                Z1 = tf.nn.conv2d(X,W1,strides = [1,2,2,1], padding = 'SAME')
                A1 = tf.nn.elu(Z1)
                P1 = tf.nn.max_pool(A1,ksize = [1, 8, 8,1], strides = [1,8,8,1],padding


                Z2 = tf.nn.conv2d(P1, W2, strides=[1,2,2, 1], padding='SAME')
                A2 = tf.nn.elu(Z2)#relu(Z2)
                P2 = tf.nn.max_pool(A2, ksize = [1, 4,4,1], strides = [1, 2,2, 1], paddi

                Z3 = tf.nn.conv2d(P2, W3, strides=[1, 1, 1, 1], padding='SAME')
                A3 = tf.nn.elu(Z3)
                P3 = tf.nn.max_pool(A3, ksize = [1, 2,2, 1], strides = [1, 2,2, 1], padd

                #W4
                Z4 = tf.nn.conv2d(P3, W4, strides=[1, 2, 2, 1], padding='SAME')
                A4 = tf.nn.elu(Z4)
                P4 = tf.nn.max_pool(A4, ksize = [1, 4,4, 1], strides = [1, 2,2, 1], padd

                #W5
                Z5 = tf.nn.conv2d(P4, W5, strides=[1,2, 2, 1], padding='SAME')
                A5 = tf.nn.elu(Z5)
                P5 = tf.nn.max_pool(A5, ksize = [1, 2,2, 1], strides = [1, 2,2, 1], padd

                # FLATTEN
                P = tf.contrib.layers.flatten(P5)
                Z6 = tf.contrib.layers.fully_connected(P, 12, activation_fn=None)
                Z7 = tf.contrib.layers.fully_connected(Z6, 12, activation_fn=None)

        return Z7
```

```
In [2]: #saver = tf.train.Saver()
        def model(X_train, Y_train, X_test, Y_test,learning_rate=0.009,
                  num_epochs=100, minibatch_size=64, print_cost=True, large_files = Fals

            tf.reset_default_graph()
            print("Batch Size : {}\nEpochs: {}\nLearning Rate: {}\nVAT: {}\nLarge_Files: {}
            title = "elu activations lr "+ str(learning_rate)+" mbs "+str(minibatch_size)+"
            if large_files:
                    title = "large images"+str(title)
            if VAT:
                    title = "VAT "+str(title)
            ops.reset_default_graph()                        # to be able to rerun the mode
```

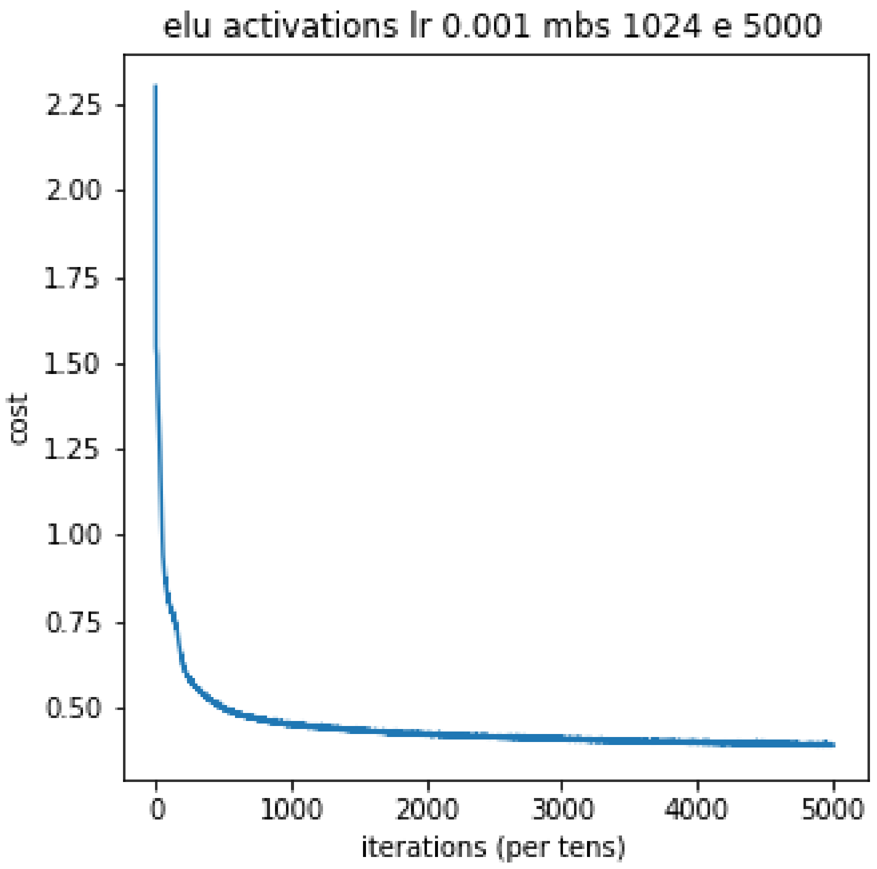
```
tf.set_random_seed(1)                                    # to keep results consistent (
seed = 3                                                 # to keep results consistent (
(m, n_H0, n_W0, n_C0) = X_train.shape
n_y = Y_train.shape[1]
costs = []                                               # To keep track of the cost

# Create Placeholders of the correct shape
X, Y = create_placeholders(n_H0, n_W0, n_C0, n_y)

# Initialize parameters
parameters = initialize_parameters()

# Forward propagation: Build the forward propagation in the tensorflow graph
Z7 = forward_propagation(X, parameters)

# Cost function: Add cost function to tensorflow graph
#cost = compute_cost(Z3, Y)
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits = Z7, label
with tf.name_scope('Optimizer'):
# Backpropagation: Define the tensorflow optimizer. Use an AdamOptimizer that mi
        optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize

#saver = tf.train.Saver()
#summary_op = tf.summary.merge_all()
config = tf.ConfigProto(allow_soft_placement = True,log_device_placement = False
config.gpu_options.allow_growth = True
sess = tf.Session(config = config)
init = tf.global_variables_initializer()
merged = tf.summary.merge([tf.summary.scalar('cross_entropy', cost)])
#writer = tf.summary.FileWriter(os.path.join(os.getcwd(),"logs"), graph=sess.gra
with sess.as_default():
        #saver.restore(sess,str(title)+".ckpt")
        sess.run(init)
        # Do the training loop

        for epoch in range(num_epochs+1):
                start = time.time()
                minibatch_cost = 0.
                num_minibatches = int(m / minibatch_size) # number of minibatche
                batch_count = int(m/minibatch_size)
                seed = seed + 1
                minibatches = random_mini_batches(X_train, Y_train, minibatch_si
                #c = 0
                for minibatch in minibatches:
                        (minibatch_X, minibatch_Y) = minibatch
                        _ , temp_cost,summary= sess.run([optimizer, cost,merged]

                        #c += 1
                minibatch_cost += temp_cost / num_minibatches
        #print(type(summary))
        end = time.time()
        if minibatch_cost <= 0.36:
                print("\n == end of training at epoch: {} with cost: {}
                break;
        #writer.add_summary(summary, epoch)
        if print_cost == False:
                drawProgressBar(epoch/num_epochs,barLen = 50)
        # Print the cost every epoch
        if print_cost == True:
                if num_epochs<100 and epoch % 2 == 0:
                        print ("Cost after epoch {}: {}".format(epoch, m
                elif num_epochs<1000:
                        if epoch % 10 == 0:
                        #end = time.time()
                                print ("Cost after epoch {}: {}".format(
                elif num_epochs >= 1000:
                        if epoch % 50 == 0:
                                print("Cost after epoch {}:{}".format(ep
        if print_cost == True and epoch % 1 == 0:
                costs.append(minibatch_cost)

#with open(str(title)+".txt","w") as f:
        #for i in range(len(costs)):
                #f.write(str(costs[i]))
                #f.write("\n")
# plot the cost
plt.figure(figsize = (5,5))
plt.plot(np.squeeze(costs))
plt.ylabel('cost')
plt.xlabel('iterations (per tens)')
plt.title(str(title))#"Learning rate =" + str(learning_rate))
plt.savefig((title)+".png")
plt.show()
with tf.device('/device:GPU:0'):
        # Calculate the correct predictions
        predict_op = tf.argmax(Z7, 1)
        correct_prediction = tf.equal(predict_op, tf.argmax(Y, 1))

        # Calculate accuracy on the test set
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
        print(accuracy)
        train_accuracy = accuracy.eval({X: X_train, Y: Y_train})
        test_accuracy = accuracy.eval({X: X_test, Y: Y_test})
print("Train Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)
#saver.save(sess,str(title)+".ckpt")
```

```
                    print("Saved Model at : {}.ckpt".format(str(title)))
                    return train_accuracy, test_accuracy, parameters
```

```
In [3]: print("======================")
        print("Training without VAT")
        print("======================")
        images,labels = load_train("im_train")
        train_x,test_x,train_y,test_y = split_data(images,labels, test_size = 0.2, shuffle = Tru
        print(train_x.shape,test_x.shape,train_y.shape,test_y.shape)

        #print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}".format(train_x3.
        start = time.time()
        _,_, params_vat_large = model(train_x,train_y,test_x,test_y,VAT = False, large_files = F
                                      ,learning_rate = 0.001, num_epochs = 5000, minibatch_size
        total_end = time.time()
        hrs = 0
        mins = (total_end-start)/60
        if mins > 60:
                hrs = mins/60
                mins %= 60
        secs = (total_end-start)%60

        images,labels = load_train("im_train_large")
        train_x,test_x,train_y,test_y = split_data(images,labels, test_size = 0.2, shuffle = Tru
        print(train_x.shape,test_x.shape,train_y.shape,test_y.shape)
        #print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}".format(train_x3.
        start = time.time()
        _,_, params_vat_large = model(train_x,train_y,test_x,test_y,VAT = False, large_files = T
                                      ,learning_rate = 0.001, num_epochs = 5000, minibatch_size
        total_end = time.time()
        hrs = 0
        mins = (total_end-start)/60
        if mins > 60:
                hrs = mins/60
                mins %= 60
        secs = (total_end-start)%60
        print("Total time taken = %i hours, %i minutes and %.4f seconds"%(hrs,mins, secs))


        print("======================")
        print("Beginning VAT")
        print("======================")
        images,labels = load_train("im_train")
        train_x,test_x,train_y,test_y = split_data(images,labels, test_size = 0.2, shuffle = Tru
        print(train_x.shape,test_x.shape,train_y.shape,test_y.shape)
        train_x2 = np.add(np.random.randn(train_x.shape[0],train_x.shape[1],train_x.shape[2],tra
        train_y2 = train_y
        train_y3 = np.append(train_y,train_y2, axis = 0)
        train_x3 = np.append(train_x,train_x2,axis = 0)
        print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}".format(train_x3.s
        start = time.time()
        _,_, params_vat_large = model(train_x3,train_y3,test_x,test_y,VAT = True, large_files =
                                      ,learning_rate = 0.001, num_epochs = 5000, minibatch_size
        total_end = time.time()
        hrs = 0
        mins = (total_end-start)/60
        if mins > 60:
                hrs = mins/60
                mins %= 60
        secs = (total_end-start)%60
        print("\n\n\nvat large files\n\n")
        images,labels = load_train("im_train_large")
        train_x,test_x,train_y,test_y = split_data(images,labels, test_size = 0.2, shuffle = Tru
        print(train_x.shape,test_x.shape,train_y.shape,test_y.shape)
        train_x2 = np.add(np.random.randn(train_x.shape[0],train_x.shape[1],
                                          train_x.shape[2],train_x.shape[3])*np.std(train_x)*0.0
        train_y2 = train_y
        train_y3 = np.append(train_y,train_y2, axis = 0)
        train_x3 = np.append(train_x,train_x2,axis = 0)
        print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}".format(train_x3.s
        start = time.time()
        _,_, params_vat_large = model(train_x3,train_y3,test_x,test_y,VAT = True, large_files =
                                      ,learning_rate = 0.001, num_epochs = 5000, minibatch_size
        total_end = time.time()
        hrs = 0
        mins = (total_end-start)/60
        if mins > 60:
                hrs = mins/60
                mins %= 60
        secs = (total_end-start)%60
        print("Total time taken = %i hours, %i minutes and %.4f seconds"%(hrs,mins, secs))
```

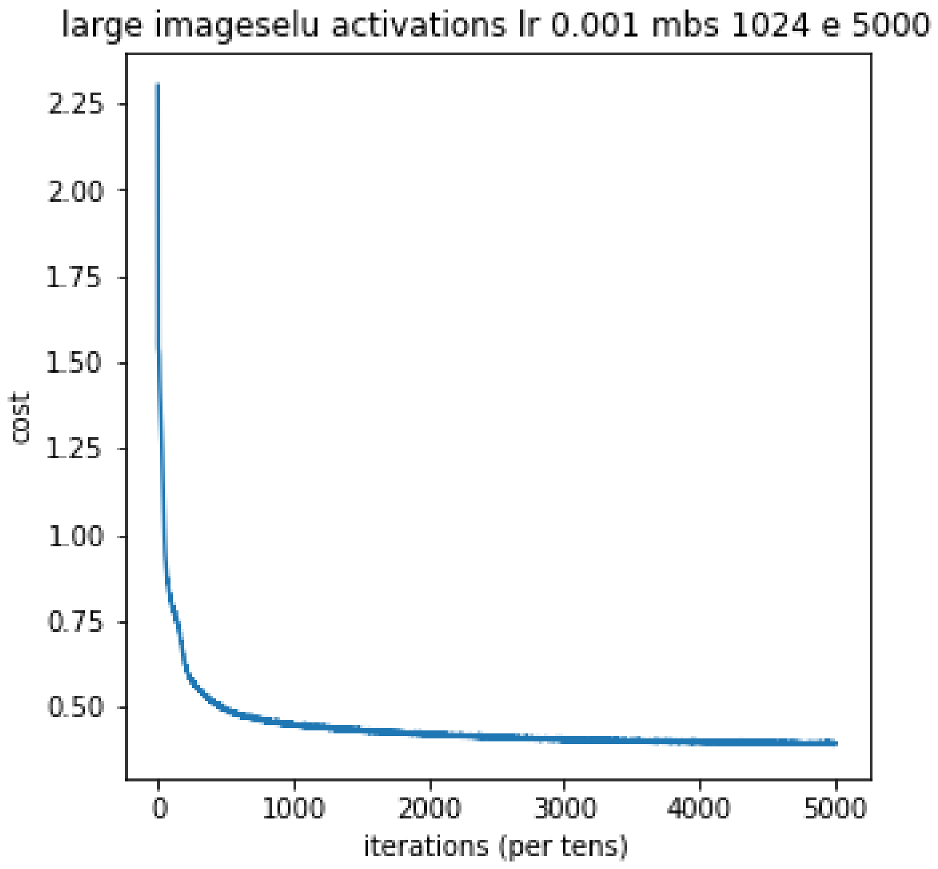
```
=====================
Training without VAT
=====================
Creating Classes, reading images and breaking things ...

[ ================================================== ] 100.00%
Done!

Done!

 images shape: (64727, 28, 28, 3), labels shape: (64727, 12)
(51781, 28, 28, 3) (12946, 28, 28, 3) (51781, 12) (12946, 12)
Batch Size : 1024
Epochs: 5000
```

```
Learning Rate: 0.001
VAT: False
Large_Files: False
Cost after epoch 0:2.301846129894256
Cost after epoch 50:0.9748922967910766
Cost after epoch 100:0.8052560436725616
Cost after epoch 150:0.7367692494392396
Cost after epoch 200:0.625106656551361
Cost after epoch 250:0.579379861354828
Cost after epoch 300:0.5580554664134979
Cost after epoch 350:0.535738506913185
Cost after epoch 400:0.5209647744894028
Cost after epoch 450:0.5062491995096208
Cost after epoch 500:0.4963218581676484
Cost after epoch 550:0.48585680365562434
Cost after epoch 600:0.4789100354909896
Cost after epoch 650:0.47274056851863866
Cost after epoch 700:0.4682585716247558
Cost after epoch 750:0.46828377068042754
Cost after epoch 800:0.46898231625556963
Cost after epoch 850:0.4574151015281678
Cost after epoch 900:0.45531943321228036
Cost after epoch 950:0.45036047577857974
Cost after epoch 1000:0.4470601963996887
Cost after epoch 1050:0.4490649497509003
Cost after epoch 1100:0.4458883184194564
Cost after epoch 1150:0.44173854947090135
Cost after epoch 1200:0.44562365233898166
Cost after epoch 1250:0.4393035316467285
Cost after epoch 1300:0.4406149768829346
Cost after epoch 1350:0.4369736832380295
Cost after epoch 1400:0.43657583534717564
Cost after epoch 1450:0.4334884989261627
Cost after epoch 1500:0.4309258526563644
Cost after epoch 1550:0.428785409927368
Cost after epoch 1600:0.42908922553062434
Cost after epoch 1650:0.42648260831832896
Cost after epoch 1700:0.4269704228639602
Cost after epoch 1750:0.42337500512599946
Cost after epoch 1800:0.4257712101936339
Cost after epoch 1850:0.42042487859725963
Cost after epoch 1900:0.42041450321674356
Cost after epoch 1950:0.4235493779182434
Cost after epoch 2000:0.42407526791095734
Cost after epoch 2050:0.4185821962356567
Cost after epoch 2100:0.4249705368280411
Cost after epoch 2150:0.4196424317359924
Cost after epoch 2200:0.4181022375822067
Cost after epoch 2250:0.4182740592956544
Cost after epoch 2300:0.4164236932992936
Cost after epoch 2350:0.415743796825409
Cost after epoch 2400:0.4115271520614623
Cost after epoch 2450:0.41823855578899366
Cost after epoch 2500:0.4134957587718963
Cost after epoch 2550:0.40982783019542696
Cost after epoch 2600:0.40986220479011537
Cost after epoch 2650:0.40793071687221527
Cost after epoch 2700:0.4106741249561309
Cost after epoch 2750:0.4093060874938965
Cost after epoch 2800:0.40613012790679925
Cost after epoch 2850:0.4131707406044005
Cost after epoch 2900:0.4102520716190339
Cost after epoch 2950:0.40763982534408566
Cost after epoch 3000:0.40906260251998894
Cost after epoch 3050:0.4057561022043226
Cost after epoch 3100:0.4016703397035598
Cost after epoch 3150:0.40549126565456406
Cost after epoch 3200:0.4083183825016021
Cost after epoch 3250:0.4012800717353822
Cost after epoch 3300:0.40613354861736306
Cost after epoch 3350:0.4003195273876191
Cost after epoch 3400:0.4006722402572632
Cost after epoch 3450:0.40324618995189665
Cost after epoch 3500:0.4002516025304794
Cost after epoch 3550:0.3998131948709489
Cost after epoch 3600:0.40020089626312255
Cost after epoch 3650:0.40015293776988975
Cost after epoch 3700:0.39773369550704946
Cost after epoch 3750:0.3985496234893798
Cost after epoch 3800:0.39698707878589645
Cost after epoch 3850:0.39714928090572355
Cost after epoch 3900:0.39862234234809885
Cost after epoch 3950:0.39745298504829407
Cost after epoch 4000:0.3948144483566284
Cost after epoch 4050:0.39812348425388316
Cost after epoch 4100:0.3951520544290542
Cost after epoch 4150:0.394593647122383
Cost after epoch 4200:0.39710810065269464
Cost after epoch 4250:0.3965498495101928
Cost after epoch 4300:0.3930836099386216
Cost after epoch 4350:0.3922786384820938
Cost after epoch 4400:0.3938040393590927
Cost after epoch 4450:0.3946526020765303
Cost after epoch 4500:0.39383527219295505
Cost after epoch 4550:0.3953688085079192
Cost after epoch 4600:0.3923367607593536
Cost after epoch 4650:0.39787890434265144
Cost after epoch 4700:0.3882081145048142
Cost after epoch 4750:0.39171828269958503
Cost after epoch 4800:0.3885872763395309
Cost after epoch 4850:0.3923960238695145
Cost after epoch 4900:0.3895100706815718
Cost after epoch 4950:0.38828426063060756
Cost after epoch 5000:0.38977077364921575
```



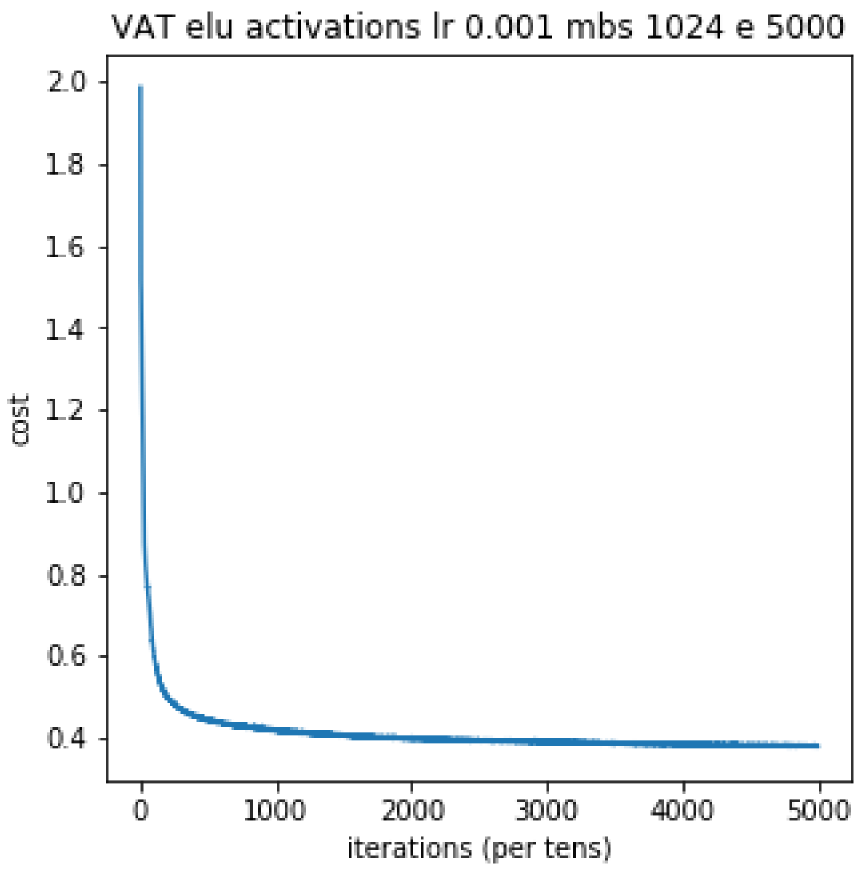
```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.877098
Test Accuracy: 0.86575
Saved Model at : elu activations lr 0.001 mbs 1024 e 5000.ckpt
Creating Classes, reading images and breaking things ...

[ ================================================== ] 100.00%
Done!

Done!
```

```
 images shape: (64727, 28, 28, 3), labels shape: (64727, 12)
(51781, 28, 28, 3) (12946, 28, 28, 3) (51781, 12) (12946, 12)
Batch Size : 1024
Epochs: 5000
Learning Rate: 0.001
VAT: False
Large_Files: True
Cost after epoch 0:2.3018461275100703
Cost after epoch 50:0.9740914690494539
Cost after epoch 100:0.805042268037796
Cost after epoch 150:0.7365570783615114
Cost after epoch 200:0.6244137036800385
Cost after epoch 250:0.5750450253486632
Cost after epoch 300:0.5533583784103393
Cost after epoch 350:0.5304897981882094
Cost after epoch 400:0.5153659802675247
Cost after epoch 450:0.5014945060014725
Cost after epoch 500:0.49238498926162727
Cost after epoch 550:0.4822921866178513
Cost after epoch 600:0.4761145114898683
Cost after epoch 650:0.47033734679222106
Cost after epoch 700:0.4668655949831009
Cost after epoch 750:0.46676043033599857
Cost after epoch 800:0.4659810030460357
Cost after epoch 850:0.45559414505958545
Cost after epoch 900:0.45371692359447485
Cost after epoch 950:0.45006166160106653
Cost after epoch 1000:0.44694679975509644
Cost after epoch 1050:0.4481680786609649
Cost after epoch 1100:0.44513563275337215
Cost after epoch 1150:0.44099114716053006
Cost after epoch 1200:0.44361104726791367
Cost after epoch 1250:0.43786072611808785
Cost after epoch 1300:0.4394246631860732
Cost after epoch 1350:0.4358276528120041
Cost after epoch 1400:0.4341731888055802
Cost after epoch 1450:0.43302676022052766
Cost after epoch 1500:0.4305608999729155
Cost after epoch 1550:0.42895447909831996
Cost after epoch 1600:0.42996555089950567
Cost after epoch 1650:0.42738989651203163
Cost after epoch 1700:0.426482400894165
Cost after epoch 1750:0.42373084127902977
Cost after epoch 1800:0.425629306435585
Cost after epoch 1850:0.42088769555091843
Cost after epoch 1900:0.420330485701561
Cost after epoch 1950:0.4230189627408981
Cost after epoch 2000:0.4232350623607636
Cost after epoch 2050:0.4186612588167191
Cost after epoch 2100:0.4233739346265793
Cost after epoch 2150:0.4184853464365006
Cost after epoch 2200:0.41731911420822143
Cost after epoch 2250:0.41692342042922975
Cost after epoch 2300:0.4154388540983202
Cost after epoch 2350:0.41480356335639973
Cost after epoch 2400:0.41102704048156746
Cost after epoch 2450:0.4172941118478775
Cost after epoch 2500:0.41325111210346227
Cost after epoch 2550:0.410364429950714
Cost after epoch 2600:0.4088292962312699
Cost after epoch 2650:0.4071749347448349
Cost after epoch 2700:0.4098604041337968
Cost after epoch 2750:0.40906790256500236
Cost after epoch 2800:0.4058103996515273
Cost after epoch 2850:0.40955744385719295
Cost after epoch 2900:0.410397926568985
Cost after epoch 2950:0.4059617638587953
Cost after epoch 3000:0.40662324309349046
Cost after epoch 3050:0.4043386048078537
Cost after epoch 3100:0.40164248526096347
Cost after epoch 3150:0.4045771878957748
Cost after epoch 3200:0.4072287267446518
Cost after epoch 3250:0.40130393743515036
Cost after epoch 3300:0.4067273694276809
Cost after epoch 3350:0.4005414307117462
Cost after epoch 3400:0.4008228093385696
Cost after epoch 3450:0.4020099419355392
Cost after epoch 3500:0.3999191898107529
Cost after epoch 3550:0.4004750561714172
Cost after epoch 3600:0.40084154427051544
Cost after epoch 3650:0.402387502193451
Cost after epoch 3700:0.39841936647892007
Cost after epoch 3750:0.3988722050189971
Cost after epoch 3800:0.3993190169334412
Cost after epoch 3850:0.39744786083698286
Cost after epoch 3900:0.39951611697673794
Cost after epoch 3950:0.3980606216192246
Cost after epoch 4000:0.3968936842679977
Cost after epoch 4050:0.39848940193653115
Cost after epoch 4100:0.3968986481428145
Cost after epoch 4150:0.3963454759120941
Cost after epoch 4200:0.399663987159729
Cost after epoch 4250:0.39897878766059874
Cost after epoch 4300:0.3951163506507875
Cost after epoch 4350:0.39486600041389475
Cost after epoch 4400:0.39808669209480274
Cost after epoch 4450:0.394076976776123
Cost after epoch 4500:0.39684869945049284
Cost after epoch 4550:0.3971363854408263
Cost after epoch 4600:0.3954790073633194
Cost after epoch 4650:0.39752203464508057
Cost after epoch 4700:0.39186489641666417
Cost after epoch 4750:0.3923180282115937
Cost after epoch 4800:0.39352940976619716
Cost after epoch 4850:0.39411123752593985
Cost after epoch 4900:0.3923121643066406
Cost after epoch 4950:0.3903877902030945
Cost after epoch 5000:0.39376635968685153
```



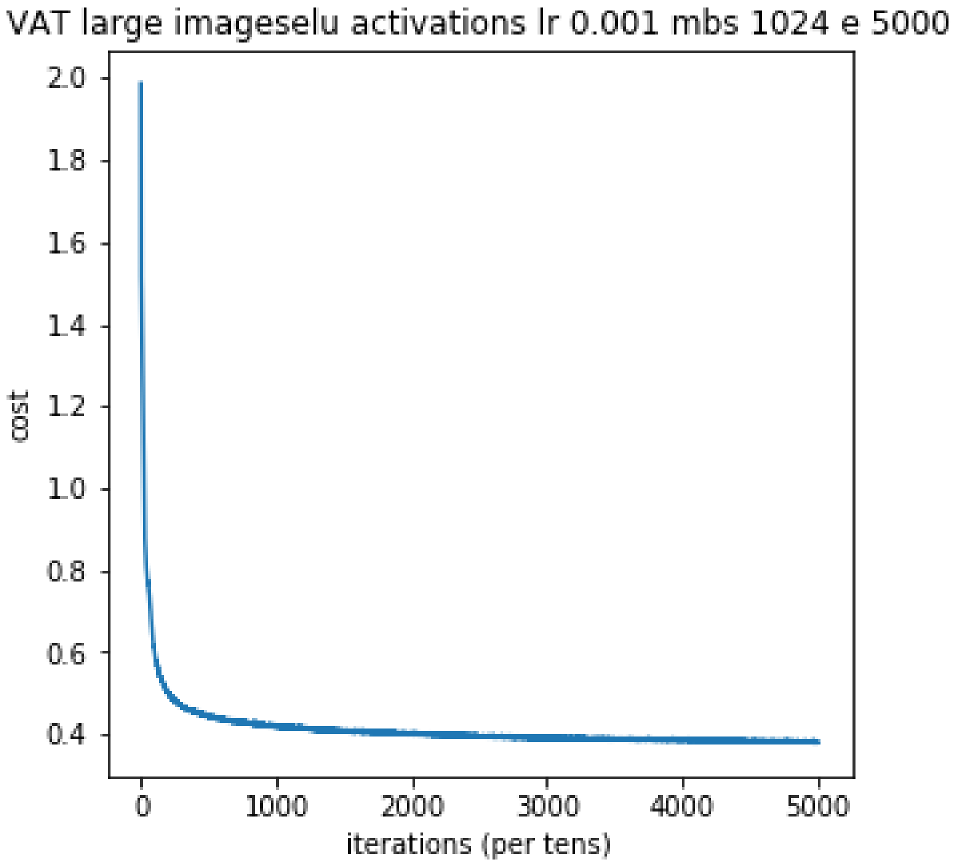
```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.878353
Test Accuracy: 0.867295
Saved Model at : large imageselu activations lr 0.001 mbs 1024 e 5000.ckpt
Total time taken = 0 hours, 50 minutes and 35.9484 seconds
```

```
=====================
Beginning VAT
=====================
Creating Classes, reading images and breaking things ...

[ ==================================================== ] 100.00%
Done!

Done!

 images shape: (64727, 28, 28, 3), labels shape: (64727, 12)
(51781, 28, 28, 3) (12946, 28, 28, 3) (51781, 12) (12946, 12)
train_x3: (103562, 28, 28, 3)
train_y3: (103562, 12)
test_x.shape: (12946, 28, 28, 3)
test_y.shape: (12946, 12)
Batch Size : 1024
Epochs: 5000
Learning Rate: 0.001
VAT: True
Large_Files: False
Cost after epoch 0:1.9863636481879956
Cost after epoch 50:0.7743765640966962
Cost after epoch 100:0.6042243903226191
Cost after epoch 150:0.5314740553350731
Cost after epoch 200:0.4990882947303282
Cost after epoch 250:0.4816551869458492
Cost after epoch 300:0.4708080687145196
Cost after epoch 350:0.4596988552867776
Cost after epoch 400:0.4553897277553484
Cost after epoch 450:0.4482005542457695
Cost after epoch 500:0.44303889764417514
Cost after epoch 550:0.4374221834805932
Cost after epoch 600:0.43559571540001607
Cost after epoch 650:0.43307782193221667
Cost after epoch 700:0.42933352188308654
Cost after epoch 750:0.43056518223026014
Cost after epoch 800:0.4243556231555372
Cost after epoch 850:0.42383685708045965
Cost after epoch 900:0.429523338480751
Cost after epoch 950:0.42435584800078124
Cost after epoch 1000:0.41956771334799203
Cost after epoch 1050:0.41684472472360823
Cost after epoch 1100:0.4175070600934549
Cost after epoch 1150:0.41214263970308945
Cost after epoch 1200:0.4154030899010082
Cost after epoch 1250:0.41213483798621936
Cost after epoch 1300:0.4083238786990099
Cost after epoch 1350:0.4098491379530123
Cost after epoch 1400:0.40657031063986304
Cost after epoch 1450:0.4059017067498499
Cost after epoch 1500:0.4065520733889966
Cost after epoch 1550:0.4061740896489361
Cost after epoch 1600:0.40661560810438474
Cost after epoch 1650:0.40289992035025407
Cost after epoch 1700:0.40143270982374046
Cost after epoch 1750:0.4020991738479916
Cost after epoch 1800:0.4017778322248176
Cost after epoch 1850:0.3988019101100391
Cost after epoch 1900:0.39897095183334746
Cost after epoch 1950:0.3983919638218265
Cost after epoch 2000:0.39889450799120535
Cost after epoch 2050:0.39659206642962924
Cost after epoch 2100:0.40157567628539453
Cost after epoch 2150:0.39883961919510724
Cost after epoch 2200:0.39767332035716224
Cost after epoch 2250:0.3958984698989603
Cost after epoch 2300:0.3962189015185479
Cost after epoch 2350:0.39320612130778854
Cost after epoch 2400:0.3926855715194552
Cost after epoch 2450:0.3926625709132394
Cost after epoch 2500:0.3921788311240696
Cost after epoch 2550:0.39203541939801495
Cost after epoch 2600:0.3944941862384872
Cost after epoch 2650:0.392931277799134
Cost after epoch 2700:0.3934823658206675
Cost after epoch 2750:0.3908986708905437
Cost after epoch 2800:0.3911950709206044
Cost after epoch 2850:0.39475105511079905
Cost after epoch 2900:0.39135416014359736
Cost after epoch 2950:0.39061896576739796
Cost after epoch 3000:0.3901975376181084
Cost after epoch 3050:0.38997258437742083
Cost after epoch 3100:0.39022557835767774
Cost after epoch 3150:0.3907653116943813
Cost after epoch 3200:0.39058733871667695
Cost after epoch 3250:0.39137356499634196
Cost after epoch 3300:0.3874336392572611
Cost after epoch 3350:0.38782413554663675
Cost after epoch 3400:0.3862891973245263
Cost after epoch 3450:0.3857378469835415
Cost after epoch 3500:0.38906942175166437
Cost after epoch 3550:0.38558001358910393
Cost after epoch 3600:0.3874575135731462
Cost after epoch 3650:0.38595532957870193
Cost after epoch 3700:0.38596622572086814
Cost after epoch 3750:0.38492031970826734
Cost after epoch 3800:0.3844660936605813
Cost after epoch 3850:0.3859408698459663
Cost after epoch 3900:0.3837019686061558
Cost after epoch 3950:0.3845447591035671
Cost after epoch 4000:0.3827491331808638
Cost after epoch 4050:0.38321367201238565
Cost after epoch 4100:0.3838783859616458
Cost after epoch 4150:0.3827747186221696
Cost after epoch 4200:0.38270848694414183
Cost after epoch 4250:0.38647168697697115
Cost after epoch 4300:0.37992781105608053
Cost after epoch 4350:0.3834785462015926
Cost after epoch 4400:0.3806133467962246
Cost after epoch 4450:0.38277064101530783
Cost after epoch 4500:0.38283245427773754
Cost after epoch 4550:0.38125467270907787
Cost after epoch 4600:0.3803724637716124
Cost after epoch 4650:0.3832058201331902
Cost after epoch 4700:0.3793182488125149
Cost after epoch 4750:0.3797380605546555
Cost after epoch 4800:0.38156418487577143
Cost after epoch 4850:0.3811866759073617
Cost after epoch 4900:0.38213621773342105
Cost after epoch 4950:0.379616400983074
Cost after epoch 5000:0.3792975783938228
```



```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.719849
Test Accuracy: 0.868531
Saved Model at : VAT elu activations lr 0.001 mbs 1024 e 5000.ckpt



vat large files


Creating Classes, reading images and breaking things ...

[ ================================================== ] 100.00%
Done!

Done!

 images shape: (64727, 28, 28, 3), labels shape: (64727, 12)
(51781, 28, 28, 3) (12946, 28, 28, 3) (51781, 12) (12946, 12)
train_x3: (103562, 28, 28, 3)
```

```
train_y3: (103562, 12)
test_x.shape: (12946, 28, 28, 3)
test_y.shape: (12946, 12)
Batch Size : 1024
Epochs: 5000
Learning Rate: 0.001
VAT: True
Large_Files: True
Cost after epoch 0:1.9863693206617148
Cost after epoch 50:0.7765667898820179
Cost after epoch 100:0.6037771389035895
Cost after epoch 150:0.5351402488085304
Cost after epoch 200:0.5015666918589338
Cost after epoch 250:0.48211624244652174
Cost after epoch 300:0.4698003725250169
Cost after epoch 350:0.46040383630459836
Cost after epoch 400:0.457314912635501
Cost after epoch 450:0.4494707610937629
Cost after epoch 500:0.4443871856916069
Cost after epoch 550:0.43970484308677144
Cost after epoch 600:0.43629541196445437
Cost after epoch 650:0.43370962467524066
Cost after epoch 700:0.4304021561499869
Cost after epoch 750:0.43210078525071094
Cost after epoch 800:0.4254985457599755
Cost after epoch 850:0.4245117743416588
Cost after epoch 900:0.4295350761696842
Cost after epoch 950:0.42671093020108664
Cost after epoch 1000:0.4200822371657532
Cost after epoch 1050:0.4188962732211198
Cost after epoch 1100:0.4181658047260624
Cost after epoch 1150:0.41360367996857883
Cost after epoch 1200:0.41812360847350405
Cost after epoch 1250:0.41477626090002534
Cost after epoch 1300:0.41029844130619936
Cost after epoch 1350:0.41125324516013123
Cost after epoch 1400:0.4081160948418153
Cost after epoch 1450:0.408755215382812
Cost after epoch 1500:0.40883886961653687
Cost after epoch 1550:0.4083147482706769
Cost after epoch 1600:0.410468066092765
Cost after epoch 1650:0.4059127669523257
Cost after epoch 1700:0.4057758128879094
Cost after epoch 1750:0.4068763825562922
Cost after epoch 1800:0.4028468323816168
Cost after epoch 1850:0.4008080484253347
Cost after epoch 1900:0.4029679903299502
Cost after epoch 1950:0.40269087859899694
Cost after epoch 2000:0.4017876129929382
Cost after epoch 2050:0.40028768718832797
Cost after epoch 2100:0.40164531605078446
Cost after epoch 2150:0.4027747638744883
Cost after epoch 2200:0.39923383369304166
Cost after epoch 2250:0.4000772644977759
Cost after epoch 2300:0.3975681202246411
Cost after epoch 2350:0.3946927838986463
Cost after epoch 2400:0.3947729019835443
Cost after epoch 2450:0.3947361746046803
Cost after epoch 2500:0.39445319417679653
Cost after epoch 2550:0.39423686679046926
Cost after epoch 2600:0.39481173145889026
Cost after epoch 2650:0.3946753359667144
Cost after epoch 2700:0.39335928075384363
Cost after epoch 2750:0.3921214477850658
Cost after epoch 2800:0.39330841586141313
Cost after epoch 2850:0.39487263796353095
Cost after epoch 2900:0.39085276174073164
Cost after epoch 2950:0.39321837950460986
Cost after epoch 3000:0.39040087886375974
Cost after epoch 3050:0.39147976867043144
Cost after epoch 3100:0.39224859688541674
Cost after epoch 3150:0.39238088437826324
Cost after epoch 3200:0.3905710678289431
Cost after epoch 3250:0.39250661860598207
Cost after epoch 3300:0.38875565611489926
Cost after epoch 3350:0.39003655520996244
Cost after epoch 3400:0.3891464412802517
Cost after epoch 3450:0.3868038117295445
Cost after epoch 3500:0.3908874867576183
Cost after epoch 3550:0.38874410461671294
Cost after epoch 3600:0.3916505023394482
Cost after epoch 3650:0.3880536491327945
Cost after epoch 3700:0.38661180716930066
Cost after epoch 3750:0.3867147756685125
Cost after epoch 3800:0.38699759174101434
Cost after epoch 3850:0.3890657719999259
Cost after epoch 3900:0.3877145382437376
Cost after epoch 3950:0.38684156743606724
Cost after epoch 4000:0.38584359890163533
Cost after epoch 4050:0.38515158838564806
Cost after epoch 4100:0.3865396244691151
Cost after epoch 4150:0.38459367710765036
Cost after epoch 4200:0.3838984715466452
Cost after epoch 4250:0.38796773287329356
Cost after epoch 4300:0.38378829678686527
Cost after epoch 4350:0.3853187944629405
Cost after epoch 4400:0.3832855386899249
Cost after epoch 4450:0.3840649667942878
Cost after epoch 4500:0.3837632649015671
Cost after epoch 4550:0.38341210089107547
Cost after epoch 4600:0.3828048818182238
Cost after epoch 4650:0.3839366577639437
Cost after epoch 4700:0.381502529772201З
Cost after epoch 4750:0.3824590111132895
Cost after epoch 4800:0.382109266106445
Cost after epoch 4850:0.3823409042145947
Cost after epoch 4900:0.383098686980729
Cost after epoch 4950:0.3812364953579289
Cost after epoch 5000:0.3803171024169072
```

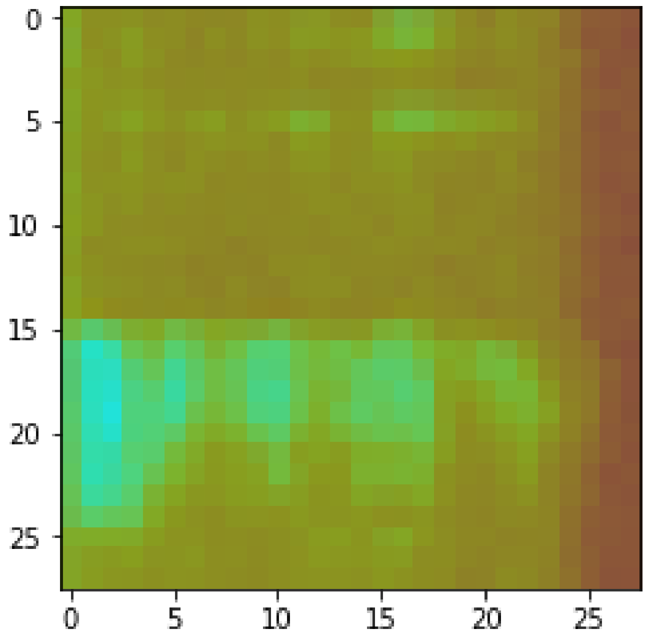


```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.718285
Test Accuracy: 0.860188
Saved Model at : VAT large imageselu activations lr 0.001 mbs 1024 e 5000.ckpt
```

```
Total time taken = 2 hours, 29 minutes and 55.3722 seconds
```

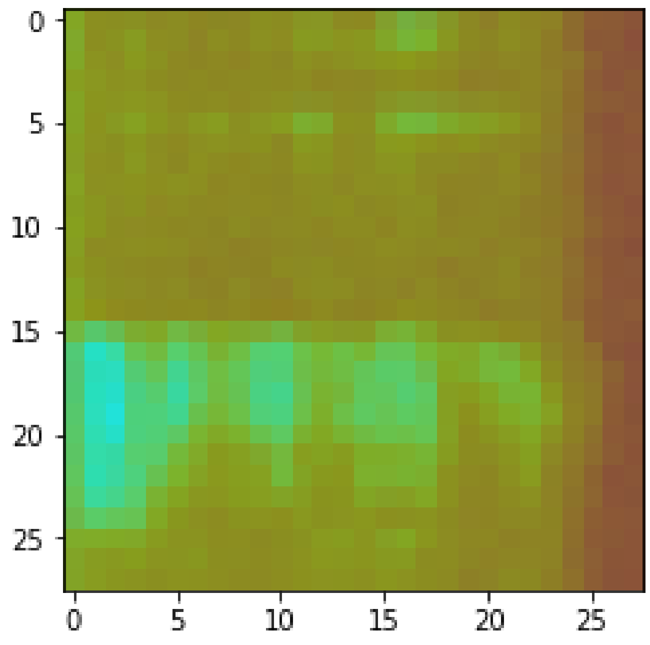
```
In [4]: plt.imshow(train_x[12])
```

```
Out[4]: <matplotlib.image.AxesImage at 0x7f0996d55d68>
```



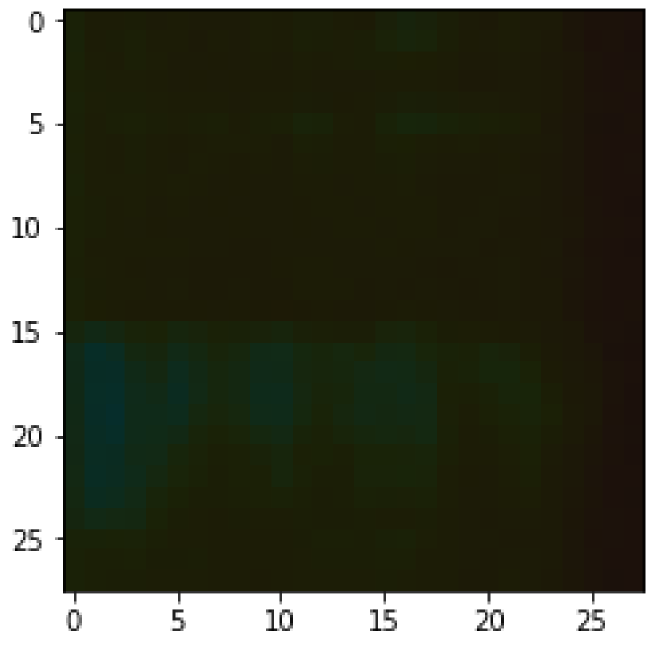
```
In [8]: plt.imshow(train_x2[12])
```

```
Out[8]: <matplotlib.image.AxesImage at 0x7f098c0af358>
```

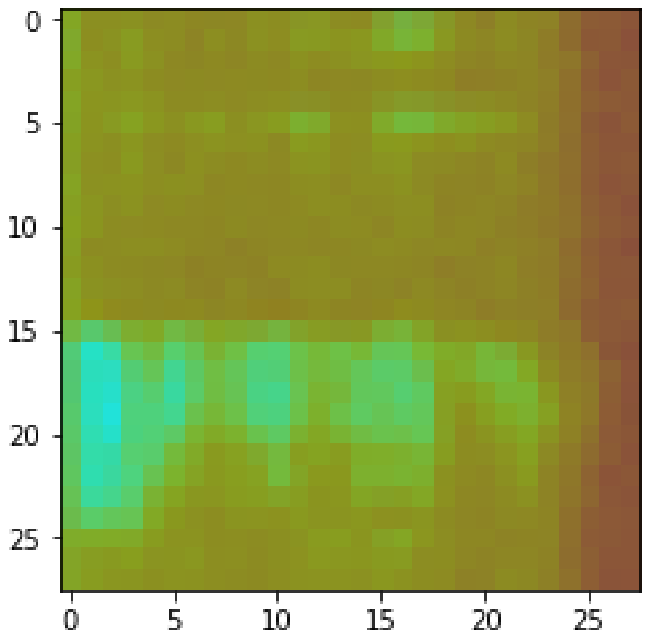


```
In [9]: print(train_y[12],train_y2[12])

[ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.] [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.
```

```
In [11]: plt.imshow(train_x[12]*np.std(train_x))

Out[11]: <matplotlib.image.AxesImage at 0x7f0996ee9a90>
```



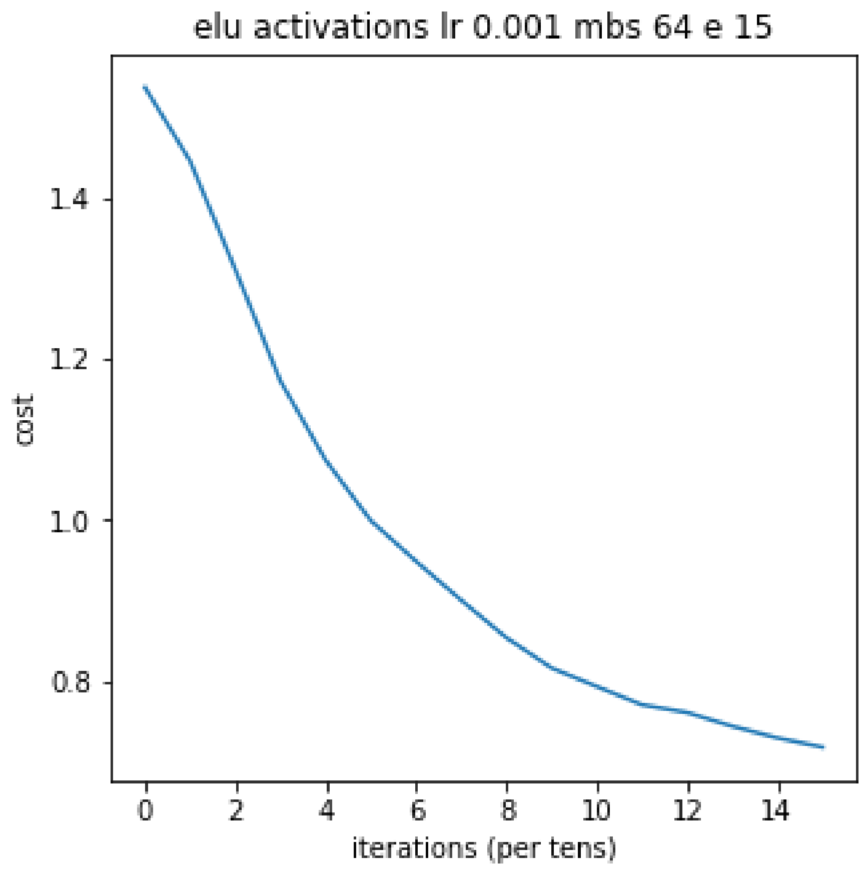
```
In [16]: plt.imshow(train_x[12]+(np.std(train_x)*0.0001))

Out[16]: <matplotlib.image.AxesImage at 0x7f0984e43978>
```



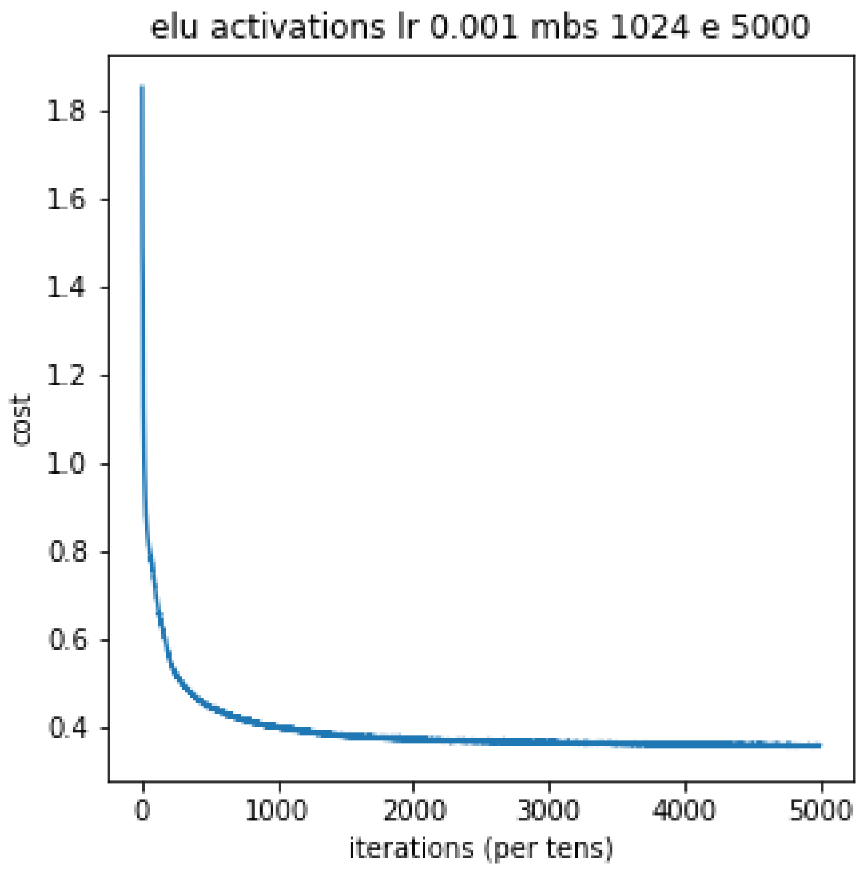
```
In [7]: train_x[12] == train_x2[12]

Out[7]: array([[[False, False, False],
                [False, False, False],
                [False, False, False],
                ...,
                [False, False, False],
                [False, False, False],
                [False, False, False]],

               [[False, False, False],
                [False, False, False],
                [False, False, False],
                ...,
                [False, False, False],
                [False, False, False],
                [False, False, False]],

               [[False, False, False],
         [False, False, False],
         [False, False, False],
         ...,
         [False, False, False],
         [False, False, False],
         [False, False, False]],

        ...,
        [[False, False, False],
         [False, False, False],
         [False, False, False],
         ...,
         [False, False, False],
         [False, False, False],
         [False, False, False]],

        [[False, False, False],
         [False, False, False],
         [False, False, False],
         ...,
         [False, False, False],
         [False, False, False],
         [False, False, False]],

        [[False, False, False],
         [False, False, False],
         [False, False, False],
         ...,
         [False, False, False],
         [False, False, False],
         [False, False, False]]], dtype=bool)
```

```
In [12]: train_x2 = np.add(np.random.randn(train_x.shape[0],train_x.shape[1],train_x.shape[2],tr
         train_y2 = train_y
         train_y3 = np.append(train_y,train_y2, axis = 0)
         train_x3 = np.append(train_x,train_x2,axis = 0)

         #print("testing on non VAT:")
         #title = "temp"
         print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}".format(train_x3.


         _,_,_ = model(train_x,train_y,test_x,test_y, learning_rate = 0.001, num_epochs = 15 , m

         #print("lr009  batch_size 32 100 epochs")
         #title = "lr 0009 e 100 mbs 2048"
         #_,_,_ = model(train_x3,train_y3,test_x,test_y, learning_rate = 0.009,num_epochs = 100,
#print("lr 0009 mbs e 1000 2048")

start = time.time()
#_, _, parameters = model(train_x3, train_y3, test_x, test_y,learning_rate=0.009,
#                   num_epochs = 1000, minibatch_size = 64, print_cost=True)
_,_, params = model(train_x3,train_y3,test_x,test_y,learning_rate = 0.001, num_epochs =
total_end = time.time()
hrs = 0
mins = (total_end-start)/60
if mins > 60:
        hrs = mins/60
        mins %= 60
secs = (total_end-start)%60
print("Total time taken = %i hours, %i minutes and %.4f seconds"%(hrs,mins, secs))
#title = "lr 0001 mb 2048 ep 5000 adv training"
#print(title)
```

```
train_x3: (103562, 28, 28, 3)
train_y3: (103562, 12)
test_x.shape: (12946, 28, 28, 3)
test_y.shape: (12946, 12)
Batch Size : 64
Epochs: 15
Learning Rate: 0.001
Cost after epoch 0: 1.538704605568178
Cost after epoch 2: 1.311808852389067
Cost after epoch 4: 1.0748123913377101
Cost after epoch 6: 0.9487186893544479
Cost after epoch 8: 0.8535776111045199
Cost after epoch 10: 0.7927363488284415
Cost after epoch 12: 0.760252226857821
Cost after epoch 14: 0.7288185738957266
```



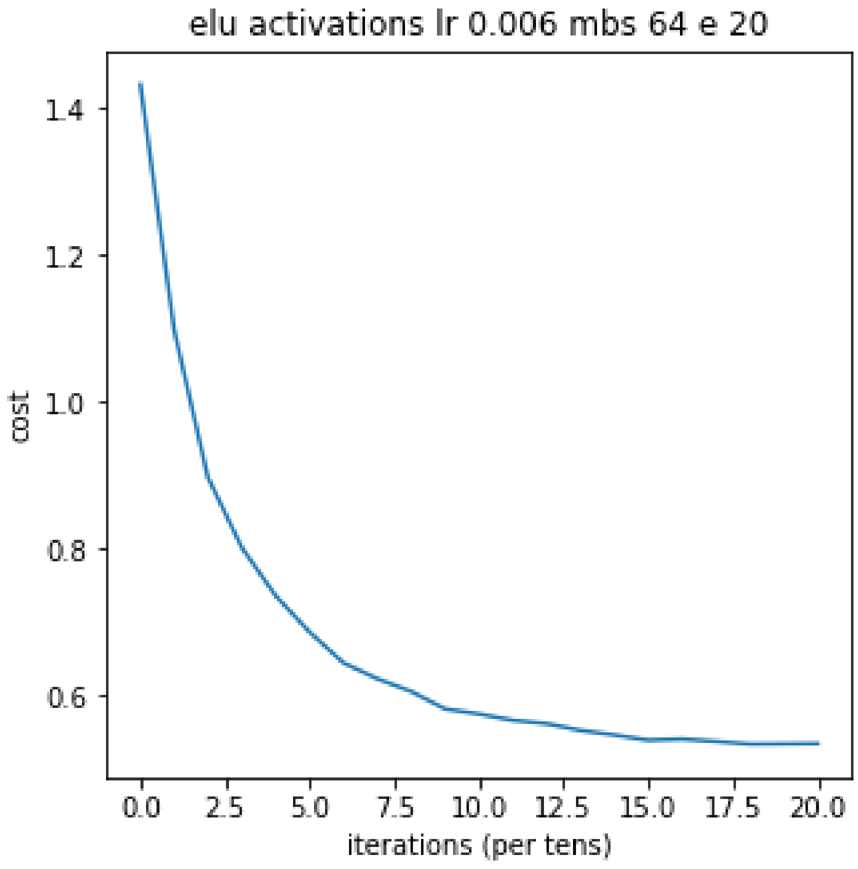
```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.774454
Test Accuracy: 0.77437
Saved Model at : elu activations lr 0.001 mbs 64 e 15.ckpt
Batch Size : 1024
Epochs: 5000
Learning Rate: 0.001
Cost after epoch 0:1.852538996403759
Cost after epoch 50:0.8171113806195778
Cost after epoch 100:0.7109046602013087
Cost after epoch 150:0.6256159585301239
Cost after epoch 200:0.5588534739938114
Cost after epoch 250:0.5184028184059825
Cost after epoch 300:0.4972533554133801
Cost after epoch 350:0.4783316788697006
Cost after epoch 400:0.46687586708824247
Cost after epoch 450:0.45240929073626457
Cost after epoch 500:0.4459702407959665
Cost after epoch 550:0.4370191713961044
Cost after epoch 600:0.43171793222427357
Cost after epoch 650:0.424940568385738
Cost after epoch 700:0.41976610327711206
Cost after epoch 750:0.41764235762086244
Cost after epoch 800:0.41072174139542156
Cost after epoch 850:0.41026092637883566
Cost after epoch 900:0.4035561102451664
Cost after epoch 950:0.40233579838630007
Cost after epoch 1000:0.4008838575665314
Cost after epoch 1050:0.3956746765882662
Cost after epoch 1100:0.39469571780450263
Cost after epoch 1150:0.3917356490498722
Cost after epoch 1200:0.39248652948011276
Cost after epoch 1250:0.38820678703855765
Cost after epoch 1300:0.384325102414235
Cost after epoch 1350:0.38503783764225424
Cost after epoch 1400:0.38150061607950975
Cost after epoch 1450:0.38100560319305643
Cost after epoch 1500:0.3817292605296219
Cost after epoch 1550:0.37757619920343466
Cost after epoch 1600:0.37868174408922106
Cost after epoch 1650:0.3757638503419291
Cost after epoch 1700:0.3757999200041932
Cost after epoch 1750:0.3761566133782415
Cost after epoch 1800:0.37555704966630077
Cost after epoch 1850:0.3734572122592738
Cost after epoch 1900:0.37479666554101626
Cost after epoch 1950:0.3739710813111598
Cost after epoch 2000:0.37201848095006274
Cost after epoch 2050:0.37151697897675023
Cost after epoch 2100:0.37148424864995605
Cost after epoch 2150:0.37145793260914267
Cost after epoch 2200:0.37041821957814813
Cost after epoch 2250:0.3677970185728359
Cost after epoch 2300:0.36807870835360906
Cost after epoch 2350:0.3673294759032751
Cost after epoch 2400:0.37016192698242617
Cost after epoch 2450:0.36923992958399343
Cost after epoch 2500:0.36606476478057337
Cost after epoch 2550:0.3688579877414325
Cost after epoch 2600:0.36671733944722923
Cost after epoch 2650:0.3671819272607862
Cost after epoch 2700:0.36778712567716537
Cost after epoch 2750:0.3641606221104613
Cost after epoch 2800:0.3667718983522735
Cost after epoch 2850:0.3646351630144782
Cost after epoch 2900:0.36538877611113074
Cost after epoch 2950:0.3638234082424994
Cost after epoch 3000:0.3634927379022731
Cost after epoch 3050:0.3642942138237528
Cost after epoch 3100:0.3655817263787335
Cost after epoch 3150:0.36339630643920146
Cost after epoch 3200:0.3622718742578335
Cost after epoch 3250:0.3619627040801663
Cost after epoch 3300:0.36469899575308995
Cost after epoch 3350:0.3630156953736105
Cost after epoch 3400:0.3611861027703427
Cost after epoch 3450:0.36110316054655767
Cost after epoch 3500:0.3655420698151731
Cost after epoch 3550:0.36290450969544974
Cost after epoch 3600:0.36264306926491247
Cost after epoch 3650:0.35925496568774234
Cost after epoch 3700:0.36193604546018165
Cost after epoch 3750:0.36161817772553695
Cost after epoch 3800:0.3589693915725934
Cost after epoch 3850:0.3611481794626406
Cost after epoch 3900:0.3593521283404661
Cost after epoch 3950:0.3586442535466488
Cost after epoch 4000:0.358838812254443
Cost after epoch 4050:0.3588472563441439
Cost after epoch 4100:0.3582562937004732
Cost after epoch 4150:0.3610579445220457
Cost after epoch 4200:0.3587377342847315
Cost after epoch 4250:0.35907701661091046
Cost after epoch 4300:0.3568122761674447
Cost after epoch 4350:0.3601248541680893
Cost after epoch 4400:0.3591436658165243
Cost after epoch 4450:0.3586022207642547
Cost after epoch 4500:0.3590317011469662
Cost after epoch 4550:0.35994430874833966
Cost after epoch 4600:0.3594999215980567
Cost after epoch 4650:0.3590289342521441
Cost after epoch 4700:0.355705211658289
Cost after epoch 4750:0.3586919558520362
Cost after epoch 4800:0.35801170308991237
Cost after epoch 4850:0.3555350879041274
Cost after epoch 4900:0.35802766798746444
Cost after epoch 4950:0.3579436449131163
Cost after epoch 5000:0.3585314610511951
```



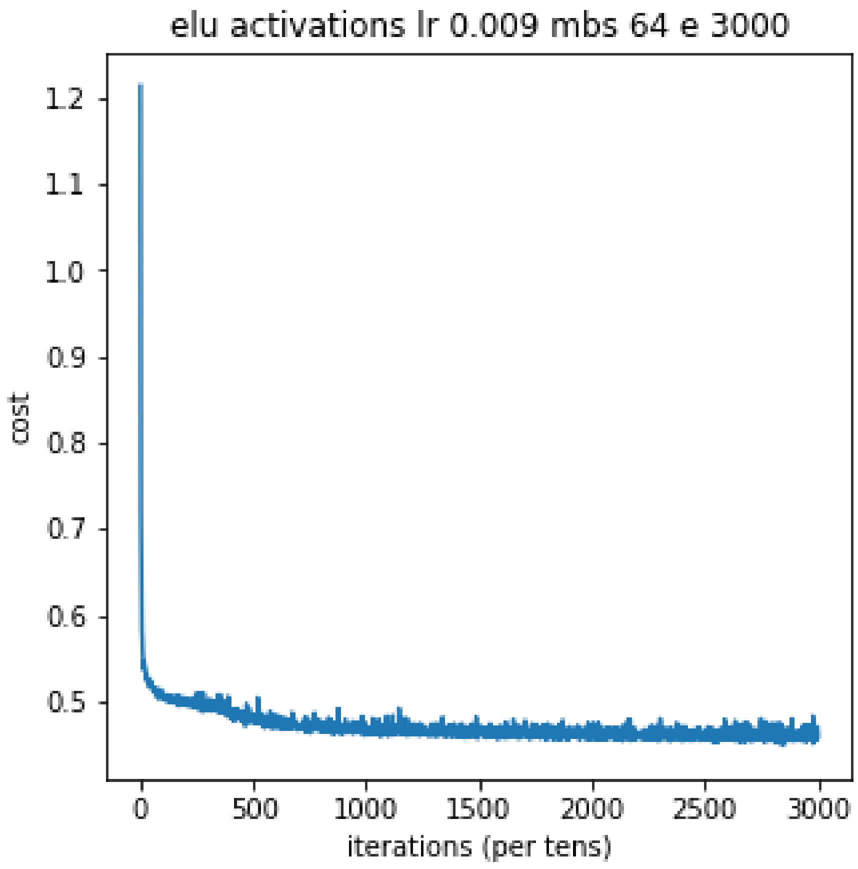
```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.716933
Test Accuracy: 0.868376
Saved Model at : elu activations lr 0.001 mbs 1024 e 5000.ckpt
Total time taken = 2 hours, 30 minutes and 5.4029 seconds
```

```
In [13]: images,labels = load_train("im_train_large")
         train_x,test_x,train_y,test_y = split_data(images,labels, test_size = 0.2, shuffle = Tr
         print(train_x.shape,test_x.shape,train_y.shape,test_y.shape)
         train_x2 = np.add(np.random.randn(train_x.shape[0],train_x.shape[1],train_x.shape[2],tr
         train_y2 = train_y
         train_y3 = np.append(train_y,train_y2, axis = 0)
         train_x3 = np.append(train_x,train_x2,axis = 0)

         #print("testing on non VAT:")
         #title = "temp"
         print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}".format(train_x3.
        _,_,_ = model(train_x,train_y,test_x,test_y, learning_rate = 0.006, num_epochs = 20 , m

        #print("lr009  batch_size 32 100 epochs")
        #title = "lr 0009 e 100 mbs 2048"
        #_,_,_ = model(train_x3,train_y3,test_x,test_y, learning_rate = 0.009,num_epochs = 100,

        #print("lr 0009 mbs e 1000 2048")

        start = time.time()
        #_, _, parameters = model(train_x3, train_y3, test_x, test_y,learning_rate=0.009,
        #                  num_epochs = 1000, minibatch_size = 64, print_cost=True)
        _,_, params = model(train_x3,train_y3,test_x,test_y,learning_rate = 0.009, num_epochs =
        total_end = time.time()
        hrs = 0
        mins = (total_end-start)/60
        if mins > 60:
                hrs = mins/60
                mins %= 60
        secs = (total_end-start)%60
        print("Total time taken = %i hours, %i minutes and %.4f seconds"%(hrs,mins, secs))
        #title = "lr 0001 mb 2048 ep 5000 adv training"
        #print(title)
```

```
Creating Classes, reading images and breaking things ...

[ ================================================== ] 100.00%
Done!

Done!

 images shape: (64727, 28, 28, 3), labels shape: (64727, 12)
(51781, 28, 28, 3) (12946, 28, 28, 3) (51781, 12) (12946, 12)
train_x3: (103562, 28, 28, 3)
train_y3: (103562, 12)
test_x.shape: (12946, 28, 28, 3)
test_y.shape: (12946, 12)
Batch Size : 64
Epochs: 20
Learning Rate: 0.006
Cost after epoch 0: 1.4328361282537248
Cost after epoch 2: 0.8969597349723871
Cost after epoch 4: 0.7359997292678933
Cost after epoch 6: 0.6447841265366603
Cost after epoch 8: 0.6057856586313954
Cost after epoch 10: 0.5750817176880146
Cost after epoch 12: 0.5619035937324888
Cost after epoch 14: 0.5466762499697272
Cost after epoch 16: 0.5410817606450598
Cost after epoch 18: 0.5342100216778004
Cost after epoch 20: 0.5348337709314309
```



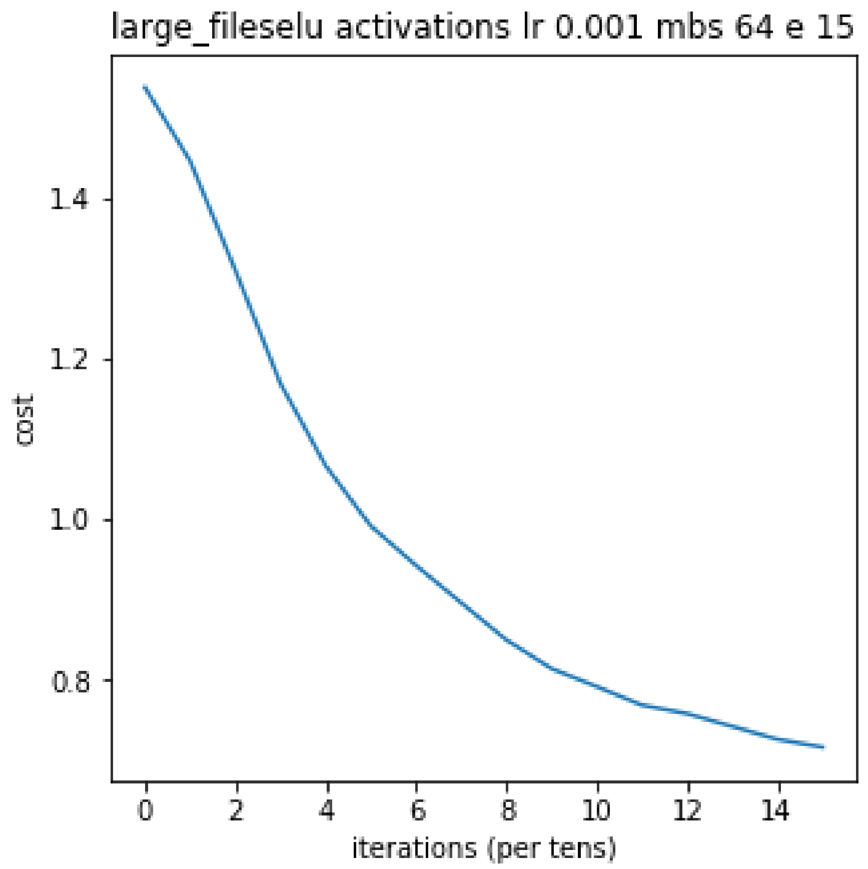
```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.817423
Test Accuracy: 0.810907
Saved Model at : elu activations lr 0.006 mbs 64 e 20.ckpt
Batch Size : 64
Epochs: 3000
Learning Rate: 0.009
Cost after epoch 0:1.214076730041628
Cost after epoch 50:0.517672833621649
Cost after epoch 100:0.5029443152644872
Cost after epoch 150:0.5031872202166822
Cost after epoch 200:0.49492061271692434
Cost after epoch 250:0.49681881143814016
Cost after epoch 300:0.49337301409730866
Cost after epoch 350:0.48941559618976727
Cost after epoch 400:0.4887893892944372
Cost after epoch 450:0.4881149947422251
Cost after epoch 500:0.48052217350865206
Cost after epoch 550:0.4791546670538096
Cost after epoch 600:0.47008518371800095
Cost after epoch 650:0.4723912452235915
Cost after epoch 700:0.47222931690431214
Cost after epoch 750:0.4706222924814822
Cost after epoch 800:0.47282864932518825
Cost after epoch 850:0.46304886661958045
Cost after epoch 900:0.4692803486923802
Cost after epoch 950:0.47443790671188857
Cost after epoch 1000:0.47029543910244576
Cost after epoch 1050:0.46352765891512177
Cost after epoch 1100:0.46789794049153305
Cost after epoch 1150:0.4730815866545348
Cost after epoch 1200:0.4638320696073935
Cost after epoch 1250:0.4625136609959359
Cost after epoch 1300:0.46606162065646933
Cost after epoch 1350:0.470956989166837
Cost after epoch 1400:0.46061257179941534
Cost after epoch 1450:0.46529506283210065
Cost after epoch 1500:0.4683816977517564
Cost after epoch 1550:0.4737128700292009
Cost after epoch 1600:0.4575453709410349
Cost after epoch 1650:0.46080240657914645
Cost after epoch 1700:0.46472604747295226
Cost after epoch 1750:0.4650317060269592
Cost after epoch 1800:0.4623086530857974
Cost after epoch 1850:0.4601124949541466
Cost after epoch 1900:0.4702517513475833
Cost after epoch 1950:0.4577486021501773
Cost after epoch 2000:0.460986780774018
Cost after epoch 2050:0.46102845158579914
Cost after epoch 2100:0.46167843579803897
Cost after epoch 2150:0.4594907573849064
Cost after epoch 2200:0.46005478545526124
Cost after epoch 2250:0.45595501197504296
Cost after epoch 2300:0.46222647383569565
Cost after epoch 2350:0.4597605670191008
Cost after epoch 2400:0.458645565584947
Cost after epoch 2450:0.4605493208368126
Cost after epoch 2500:0.4596046902727429
Cost after epoch 2550:0.458687186452971
Cost after epoch 2600:0.4642547749452584
Cost after epoch 2650:0.4587041393696008
Cost after epoch 2700:0.4574554533650463
Cost after epoch 2750:0.45807699313932987
Cost after epoch 2800:0.46820428080589677
Cost after epoch 2850:0.4555797420156611
Cost after epoch 2900:0.45437858263848036
Cost after epoch 2950:0.4630639507850846
Cost after epoch 3000:0.46820527239620885
```



```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.703868
Test Accuracy: 0.845126
Saved Model at : elu activations lr 0.009 mbs 64 e 3000.ckpt
Total time taken = 5 hours, 12 minutes and 3.5579 seconds
```

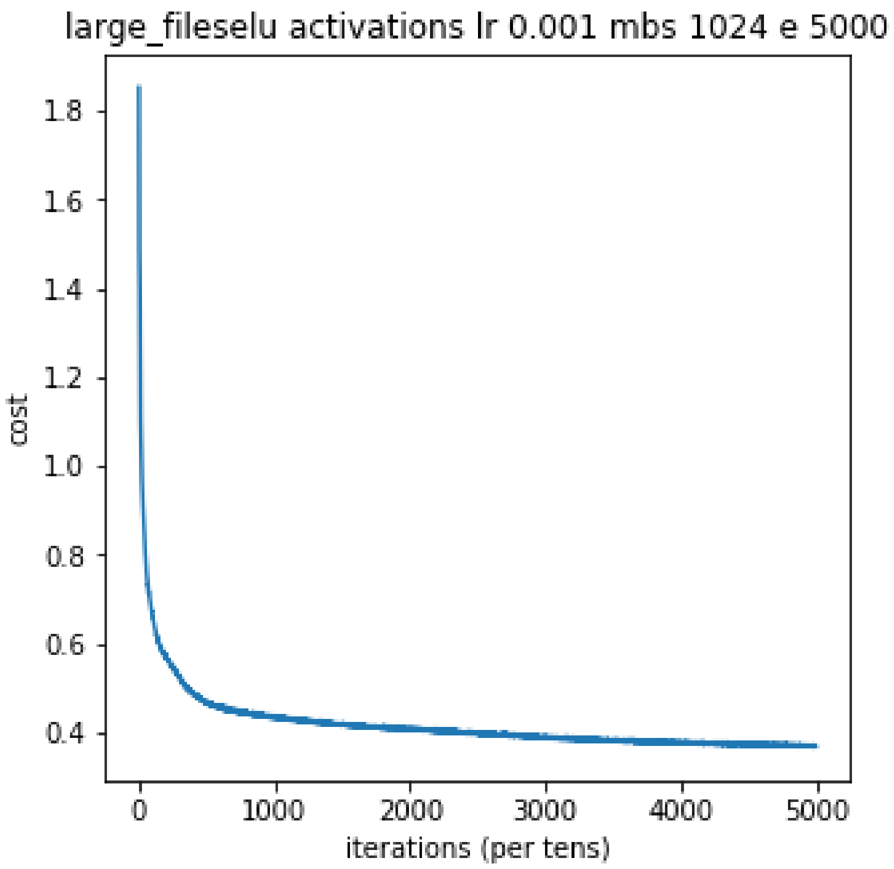
```
In [15]: #print("testing on non VAT:")
         #title = "temp"
         print("train_x3: {}\ntrain_y3: {}\ntest_x.shape: {}\ntest_y.shape: {}\n\n\n".format(tra


         _,_,_ = model(train_x,train_y,test_x,test_y, learning_rate = 0.001, num_epochs = 15 , m
        #print("lr009  batch_size 32 100 epochs")
        #title = "lr 0009 e 100 mbs 2048"
        #_,_,_ = model(train_x3,train_y3,test_x,test_y, learning_rate = 0.009,num_epochs = 100,

        #print("lr 0009 mbs e 1000 2048")
        print("\n\n\n\n\n")
        start = time.time()
        #_, _, parameters = model(train_x3, train_y3, test_x, test_y,learning_rate=0.009,
        #                   num_epochs = 1000, minibatch_size = 64, print_cost=True)
        _,_, params = model(train_x3,train_y3,test_x,test_y,learning_rate = 0.001, num_epochs =
        total_end = time.time()
        hrs = 0
        mins = (total_end-start)/60
        if mins > 60:
                hrs = mins/60
                mins %= 60
        secs = (total_end-start)%60
        print("Total time taken = %i hours, %i minutes and %.4f seconds"%(hrs,mins, secs))
        #title = "lr 0001 mb 2048 ep 5000 adv training"
        #print(title)
```

```
train_x3: (103562, 28, 28, 3)
train_y3: (103562, 12)
test_x.shape: (12946, 28, 28, 3)
test_y.shape: (12946, 12)





Batch Size : 64
Epochs: 15
Learning Rate: 0.001
Cost after epoch 0: 1.5387043952941868
Cost after epoch 2: 1.3108156644664388
Cost after epoch 4: 1.0675960644361273
Cost after epoch 6: 0.9431184435951702
Cost after epoch 8: 0.8503288585780143
Cost after epoch 10: 0.7920647936289477
Cost after epoch 12: 0.7582914373193892
Cost after epoch 14: 0.7265116017224321
```



```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.774493
Test Accuracy: 0.771821
Saved Model at : elu activations lr 0.001 mbs 64 e 15.ckpt
```

```
Batch Size : 1024
Epochs: 5000
Learning Rate: 0.001
Cost after epoch 0:1.8525937866456443
Cost after epoch 50:0.7944315276523625
Cost after epoch 100:0.6641651663449732
Cost after epoch 150:0.6009879041426256
Cost after epoch 200:0.5680660623134953
Cost after epoch 250:0.5452337896469795
Cost after epoch 300:0.526236864009706
Cost after epoch 350:0.5009702582760612
Cost after epoch 400:0.48959486791403
Cost after epoch 450:0.47548144023017114
Cost after epoch 500:0.465557597061195
Cost after epoch 550:0.4590094328516781
Cost after epoch 600:0.45751005322626326
Cost after epoch 650:0.45044090193096964
Cost after epoch 700:0.44672269632320594
Cost after epoch 750:0.45120323441996435
Cost after epoch 800:0.44180875426471844
Cost after epoch 850:0.43966622370304437
Cost after epoch 900:0.43756849665452946
Cost after epoch 950:0.43890497294983066
Cost after epoch 1000:0.43481361482403064
Cost after epoch 1050:0.43244724019919273
Cost after epoch 1100:0.4296095403704313
Cost after epoch 1150:0.4292889972134392
Cost after epoch 1200:0.4276911878939902
Cost after epoch 1250:0.42785356451969325
Cost after epoch 1300:0.42312396192314583
Cost after epoch 1350:0.4203681405818109
Cost after epoch 1400:0.4177578888317144
Cost after epoch 1450:0.41753505244113426
Cost after epoch 1500:0.41898734144645167
Cost after epoch 1550:0.4173866725794158
Cost after epoch 1600:0.4155958769935193
Cost after epoch 1650:0.412009844685545
Cost after epoch 1700:0.41263313724262884
Cost after epoch 1750:0.4103222394933796
Cost after epoch 1800:0.41275876671961037
Cost after epoch 1850:0.4085706758617172
Cost after epoch 1900:0.4106858455308592
Cost after epoch 1950:0.40949143956203266
Cost after epoch 2000:0.4079982912776493
Cost after epoch 2050:0.40718402514363267
Cost after epoch 2100:0.40739107309001515
Cost after epoch 2150:0.4071366249924839
Cost after epoch 2200:0.40736196536828956
Cost after epoch 2250:0.4015504207351421
Cost after epoch 2300:0.4032210563078966
Cost after epoch 2350:0.40193838854827507
Cost after epoch 2400:0.4022044675184949
Cost after epoch 2450:0.4000071635340699
Cost after epoch 2500:0.39780283387344645
Cost after epoch 2550:0.39969331763758525
Cost after epoch 2600:0.3975473376783996
Cost after epoch 2650:0.3959874454701301
Cost after epoch 2700:0.3954371290631814
Cost after epoch 2750:0.3918093041618272
Cost after epoch 2800:0.39400946385789604
Cost after epoch 2850:0.3924647838172345
Cost after epoch 2900:0.3922883614455119
Cost after epoch 2950:0.38909446278421
Cost after epoch 3000:0.3921494401327452
Cost after epoch 3050:0.38793401877478817
Cost after epoch 3100:0.3889636668828455
Cost after epoch 3150:0.3875336027381444
Cost after epoch 3200:0.3868750501977336
Cost after epoch 3250:0.38579984850222526
Cost after epoch 3300:0.38377507783398773
Cost after epoch 3350:0.3842131524983018
Cost after epoch 3400:0.38436458695052883
Cost after epoch 3450:0.3827552819015956
Cost after epoch 3500:0.38328423742020473
Cost after epoch 3550:0.38191566608919947
Cost after epoch 3600:0.38299691824629756
Cost after epoch 3650:0.3803006062413207
Cost after epoch 3700:0.38209043339927595
Cost after epoch 3750:0.37918011268766794
Cost after epoch 3800:0.37865967827268154
Cost after epoch 3850:0.3808071569050891
Cost after epoch 3900:0.3792786167399718
Cost after epoch 3950:0.3786873543026423
Cost after epoch 4000:0.3751646817320645
Cost after epoch 4050:0.3764845457407508
Cost after epoch 4100:0.37672676514871056
Cost after epoch 4150:0.37583859250097007
Cost after epoch 4200:0.3748282807888372
Cost after epoch 4250:0.3767845863753025
Cost after epoch 4300:0.3741174614665532
Cost after epoch 4350:0.3749482041538352
Cost after epoch 4400:0.3728360229789619
Cost after epoch 4450:0.37456899704319424
Cost after epoch 4500:0.37209699531592955
Cost after epoch 4550:0.3750002354678541
Cost after epoch 4600:0.37200161814689625
Cost after epoch 4650:0.37348023293041954
Cost after epoch 4700:0.3707474927500922
Cost after epoch 4750:0.37191988365484935
Cost after epoch 4800:0.37074655294418335
Cost after epoch 4850:0.370646751458102
Cost after epoch 4900:0.37248979700673934
Cost after epoch 4950:0.36950153407484015
Cost after epoch 5000:0.3687083098557915
```



```
Tensor("Mean_1:0", shape=(), dtype=float32, device=/device:GPU:0)
Train Accuracy: 0.718304
Test Accuracy: 0.869458
Saved Model at : elu activations lr 0.001 mbs 1024 e 5000.ckpt
Total time taken = 2 hours, 30 minutes and 4.7961 seconds
```